Amar Fuel

By

Shah Newaz Aziz 011201437
Al-Momen Reyad 011203011
Sonjoy Dey 011202074
James Anthony Purification 011201242
Jonathan George Blaize Purification 011201244
Md.Abu Rayhan Abir 011201452

Submitted in partial fulfilment of the requirements
of the degree of Bachelor of Science in Computer Science and Engineering

April 29, 2024

Department of Computer Science and Engineering
United International University

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

Drivers on the Dhaka-Chattogram highway allegedly engage in the illicit trade of fuel oil, leading to significant financial losses and security concerns for fuel station owners. Despite the prevalence of this issue, law enforcement agencies have not taken adequate action.

1.2 Motivation

The motivation behind the project is to address the rampant trade of stolen fuel oil by implementing a comprehensive automation system for fuel stations. This system aims to enhance transparency, security, and efficiency in fuel transactions.

1.3 Objectives

- Develop an IoT-based automation system for fuel stations to monitor fuel dispensers and tank levels in real-time.
- Implement robust security measures to prevent fuel theft and unauthorized access.
- Integrate online payment gateways for secure and convenient transactions.
- Provide user-friendly interfaces for both fuel station owners and customers.
- Enhance operational efficiency through automation capabilities.

1.4 Project Outcome

The project aims to revolutionize the fuel station industry in Bangladesh by transforming traditional stations into technologically advanced establishments. The outcome will be a connected ecosystem that ensures transparency, security, and efficiency in fuel transactions.

Chapter 2

Tactics

2.1 Availability

2.1.1 Detect Faults

1. **Monitor:** Continuously monitor the fuel station automation system components, including software and hardware, to detect any abnormal behavior or performance issues that could impact the availability of the system. This monitoring should cover aspects such as fuel dispenser activity, tank levels, and online/offline status.

2. **Ping/echo:** Regularly send test signals to critical system components, such as fuel dispensers and tank sensors, to ensure they are responsive and detect any network connectivity issues that could affect the availability of the system.

3. **Timestamp:** Log timestamps for all system events and transactions, including fuel sales, tank refills, and software updates, to facilitate fault detection and analysis in accordance with the timestamping feature mentioned in the scope documentation.

4. **Condition Monitoring:** Implement sensors or monitoring tools to continuously assess the condition of vital system components, like fuel dispensers and tank gauges, to detect potential faults or malfunctions before they lead to system downtime.

5. **Exception Detection:** Set up mechanisms to detect exceptions or anomalies in system operations, such as discrepancies in fuel sales data or irregularities in tank level readings, to proactively address potential availability issues.

2.1.2 Prevent Faults

1. **Remove from service:** Implement protocols to temporarily remove malfunctioning fuel dispensers or other hardware components from service to prevent disruptions to fuel station operations and ensure the availability of alternative units for customers, aligning with the system's goal of providing uninterrupted service.

2. **Transactions:** Incorporate transactional safeguards into the system to ensure the accuracy and completeness of fuel sales transactions, mitigating the risk of partial or erroneous transactions that could compromise system availability.

3. **Exception Prevention:** Integrate error handling mechanisms and validation checks into the system software to prevent exceptions or abnormalities in fuel station operations, reducing the likelihood of system downtime and ensuring continuous availability.

4. **Increase Competence Set:** Offer training programs and resources to fuel station staff on system operation, maintenance, and troubleshooting, enhancing their competence in managing and maintaining system availability as specified in the vision and scope documentation.

2.1.3 Recover from faults

1. **Reconfiguration:** Implement automated reconfiguration processes to restore system functionality in the event of a component failure, ensuring minimal downtime and maintaining availability as per the system's objectives.

2. **Software update:** Deploy timely software updates and patches to address known issues or vulnerabilities in the system, enhancing its resilience and availability while aligning with the vision of providing a technologically advanced solution.

3. **Ignore faulty behavior:** Develop fallback mechanisms or alternative workflows within the system to bypass or mitigate the impact of faulty components, ensuring continued availability of critical functions even in the presence of faults.

4. **Retry:**Incorporate retry mechanisms into the system's transactional processes to automatically retry failed operations, such as failed fuel sales transactions, to recover from transient faults and maintain system availability for customers.

2.2 Deployability

2.2.1 Manage Deployment Pipeline

1. **Scale rollouts:** A common tactic in software deployment, involves gradually implementing updates or new features. Instead of updating all users at once, changes are initially tested on a smaller subset. This controlled approach allows for assessing performance, stability, and user feedback before full deployment. For example, in our scenario, we've developed a new feature for the Fuel Point of Sale module that enables fuel station owners to offer discounts to loyal customers. We begin by deploying this feature to a small group of stations to gather feedback and ensure stability. If successful, we gradually expand its rollout. This method reduces risks by testing the feature on a smaller scale before full implementation.

2. **Rollback:** It is a crucial strategy in managing deployments. It involves undoing a deployment and reverting to the previous state if issues arise or expectations aren't met. Since deployments often include multiple changes at once, it's important for the rollback process to track all these changes. Think of it like rewinding a movie to a specific scene, but in this case, we're returning the software to its previous condition. Ideally, this process happens automatically with little manual intervention. For instance, if we deploy an update to the Customer Module and encounter unexpected errors affecting fuel ordering for credit customers, we promptly initiate a rollback to the previous version. This helps minimize downtime and customer impact while we investigate the cause of the errors.

2.2.2 Manage Deployed System

1. **Package dependencies:** Package dependencies involve bundling an element with its dependencies so that they are deployed together. This ensures that the versions of the dependencies remain consistent as the element progresses from development to production environments.

2. **Feature toggle:** This is a handy tool that lets you control new features in your system, even after they've been deployed and tested. It's like having a "kill switch" that can turn off a feature at runtime without needing a new deployment. This flexibility allows you to manage features without the cost and risk of redeploying services. In our scenario, we've developed a new feature for the Admin module that enables administrators to generate custom reports. Before rolling out this feature to all administrators, we utilize feature toggles. This means we selectively enable the feature for a subset of trusted users. By doing so, we can gather feedback and conduct additional testing before releasing the feature to all users. This approach ensures that the feature is thoroughly vetted and meets the needs of our users before being widely implemented.

2.3 Energy Efficiency

2.3.1 Monitor Resources

1. **Metering:** Metering, in the context of energy efficiency, involves implementing mechanisms to track the energy consumption of various components within the system. This includes fuel dispensers, tank monitoring systems, and IoT devices. By utilizing energy meters to measure CPU usage accurately, the system can identify areas for optimization and ensure efficient resource utilization. Metering allows for comprehensive monitoring of energy usage throughout the system, enabling informed decision-making and targeted optimization efforts to minimize energy wastage and enhance overall efficiency.

2. **Static Classification:** Static classification entails categorizing devices and components based on their energy consumption characteristics and operational patterns. By identifying high-energy-consuming components such as pumps and compressors, the system can prioritize optimization efforts to reduce energy usage in these areas. Static classification facilitates targeted optimization strategies to improve overall energy efficiency and minimize wastage.

2.3.2 Allocate Resources

1. **Reduce Usage:** To reduce usage, the system implements energy-saving modes or standby functionalities for devices during periods of low activity. By utilizing smart scheduling algorithms, non-essential components can be turned off or have their power reduced when they are not in use, effectively minimizing energy consumption without sacrificing functionality.

2. **Schedule Resources:** The system employs scheduling algorithms to optimize resource usage based on demand forecasts and operational requirements. By scheduling routine maintenance and system updates during off-peak hours, disruptions and energy consumption are minimized, ensuring efficient resource allocation and enhancing overall system efficiency.

2.3.3 Reduce Resources Demand

1. **Manage Event Arrival:** Managing event arrival involves implementing event-driven architecture to handle requests and events efficiently. By utilizing queueing systems to buffer incoming requests, the system can smooth out spikes in resource demand, reducing energy peaks and ensuring stable operation.

2. **Prioritize:** Prioritizing critical system functions and processes ensures efficient resource allocation. By implementing priority-based scheduling, resources are allocated to mission-critical tasks first, optimizing energy usage in high-demand scenarios and improving overall system reliability.

3. **Bound Execution Time:** Setting upper bounds on the execution time of tasks and processes helps limit energy consumption. By implementing timeout mechanisms to terminate resource-intensive operations that exceed predefined thresholds, energy wastage is prevented, ensuring efficient resource utilization and energy efficiency.

2.3.4 Utility tree

Quality Attribute	**Attribute Refinement**	**ASR Scenario**
Performance	Automation capabilities	Automating processes should optimize performance, ensuring swift and efficient operations at fuel stations.
	Integration with online payment gateways	Seamless integration with payment gateways should expedite transactions, minimizing wait times for customers.
Usability	User-friendly interface	Intuitive interfaces should enhance usability, making it easy for both owners and customers to navigate the system.
Maintainability	Automated data entry system	Automated data entry ensures data accuracy and reduces the need for manual intervention, simplifying maintenance.
	Integration of IoT-based infrastructure	Modular and scalable infrastructure simplifies updates and modifications, ensuring long-term maintainability.
Configurability	Device & Nozzle Setup	Configurable settings for device and nozzle setups allow customization based on station requirements.
	Product Setup	Flexible product setup enables stations to easily add, modify, or remove products as needed.
Security	Incorporation of robust security measures	Robust security measures safeguard sensitive data and prevent unauthorized access, ensuring data integrity.
Availability	No downtime	The system supports 24/7/365 access for customers.
	Automated data entry system	Automated data entry minimizes downtime due to errors, ensuring continuous availability of accurate information.

Table 2.1: Utility Tree for Quality Attributes

Chapter 3

Project Design

3.1 Requirement Analysis

3.1.1 ER Diagram

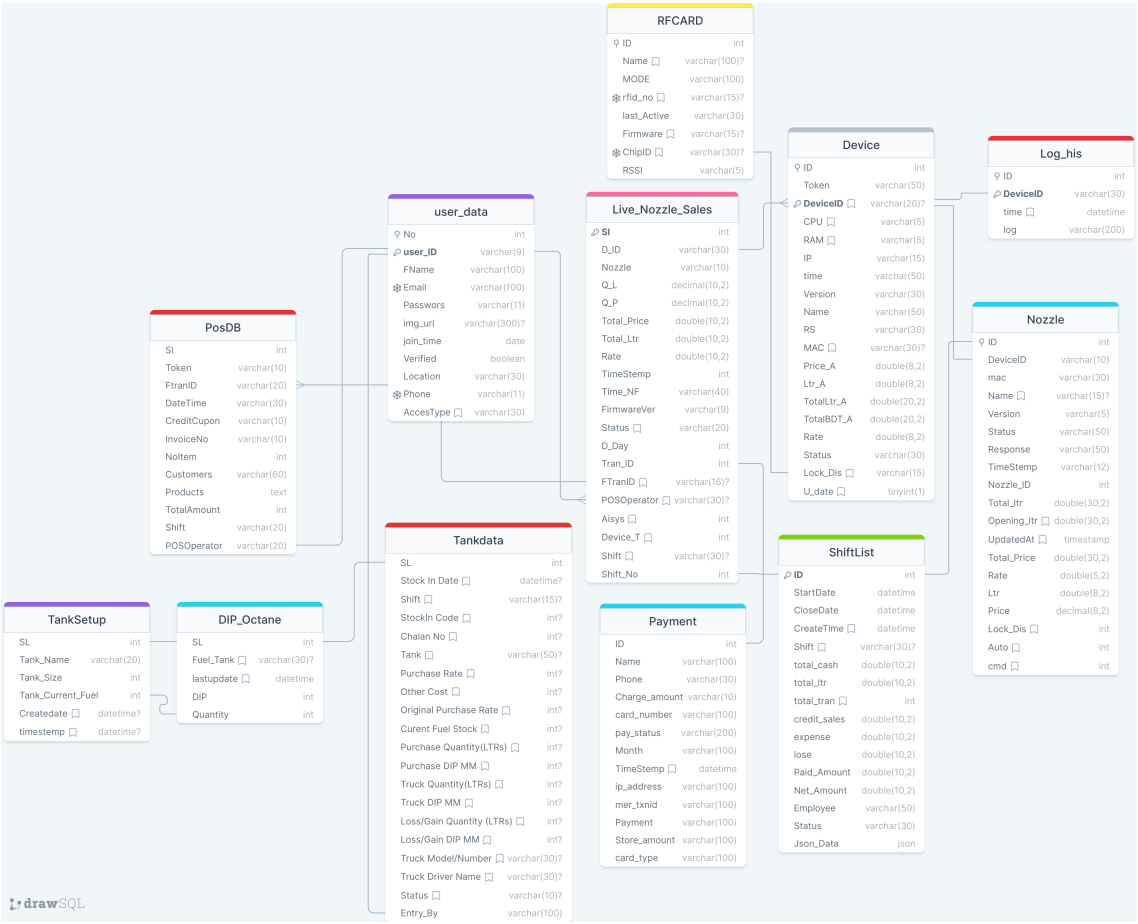

Figure 3.1: Entity-Relationship (ER) Diagram

3.1.2 Functional and Nonfunctional Requirements

3.1.3 Context Diagram

3.1.4 Data Flow Diagram Level 1

3.1.5 UI Design

3.2 Detailed Methodology and Design

You have to mention alternate solutions that you have considered. Why you have selected the specific solution, etc.

3.3 Project Plan

3.4 Task Allocation

3.5 Summary

Chapter 4

Implementation and Results

4.1 Environment Setup

4.2 Testing and Evaluation

4.3 Results and Discussion

4.4 Summary

Chapter 5

Standards and Design Constraints

[Must be present in FYDP-1 Report and also in Final Report]

Every chapter should start with 1-2 sentences on the outline of the chapter.

5.1 Compliance with the Standards

Only mention the standards that are related to your project. This list is not complete. For each of the standards discuss the alternates with pros and cons and rationale of selection.

5.1.1 Software Standards

5.1.2 Hardware Standards

5.1.3 Communication Standards

5.2 Design Constraints

Only mention the constraints that are related to the design of your project. This list is not complete.

5.2.1 Economic Constraint

5.2.2 Environmental Constraint

5.2.3 Ethical Constraint

5.2.4 Health and Safety Constraint

5.2.5 Social Constraint

5.2.6 Political Constraint

5.2.7 Sustainability

5.3 Cost Analysis

Provide a cost analysis in terms of budget required and revenue model. In case of budget, you must show an alternate budget and rationales.

5.4 Complex Engineering Problem

5.4.1 Complex Problem Solving

In this section, provide a mapping with problem solving categories. For each mapping add subsections to put rationale (Use Table 5.1). For P1, you need to put another mapping with Knowledge profile and rational thereof.

Table 5.1: Mapping with complex problem solving.

P1 Dept of Knowledge	P2 Range of Conflicting Requirements	P3 Depth of Analysis	P4 Familiarity of Issues	P5 Extent of Applicable Codes	P6 Extent of Stakeholder Involvement	P7 Interdependence
✓	✓					

5.4.2 Engineering Activities

In this section, provide a mapping with engineering activities. For each mapping add subsections to put rationale (Use Table 5.2).

5.5 Summary

Table 5.2: Mapping with complex engineering activities.

A1 Range of resources	A2 Level of Interaction	A3 Innovation	A4 Consequences for society and environment	A5 Familiarity
√	√			

Chapter 6

Conclusion

[Must be present in FYDP-1 Report and also in Final Report. Might be incomplete in FYDP-1 Report.]

Every chapter should start with 1-2 sentences on the outline of the chapter.

6.1 Summary

6.2 Limitation

6.3 Future Work

References